Problem 1. Given an array $a[1, \ldots, n]$ of integers, consider the problem of finding the longest monotonically increasing subsequence of not necessarily contiguous entries in your array. For example, if the entries are 10, 3, 9, 5, 8, 13, 11, 14, then a longest monotonically increasing subsequence is 3, 5, 8, 11, 14. The goal of this problem is to find an efficient dynamic programming algorithm for solving this problem.

HINT: Let $L[i]$ be the length of the longest monotonically increasing sequence that ends at and uses element $a[i]$.

(a) Write down a recurrence for $L[i]$.

(b) Based on your recurrence, give a recursive program for finding the length of the longest monotonically increasing subsequence. What can you say about its running time?

(c) Show how to MEMOIZE the recursive algorithm. Argue that it has $\Theta(n^2)$ running time.

(d) Based on your recurrence, give a $\Theta(n^2)$ dynamic programming algorithm for finding the length of the longest monotonically increasing subsequence.

(e) Modify your algorithm to actually find the longest increasing subsequence (not just its length).

(f) Improve your algorithm so that it finds the length of the longest monotonically increasing subsequence in $\Theta(n \log n)$ time.

Problem 2. Do Exercise 16 on page 327 of Kleinberg and Tardos. Justify your answer.

Problem 3. For the previous problem, given that there are $n$ people, what is the minimum number of rounds you need to notify everyone. Justify your answer. (How should the tree be arranged?)